

Disclaimer: *These are rough notes, with some exercises.*

18.1 Non-bipartite matching.

Question: Say we are given a graph, how do you find a maximum matching?

Well, consider an optimal matching, and a current matching.

Question: What is the difference?

A set of cycles and paths.

Question: What is the number of paths?

The difference in the size of the matchings.

Question: How do you proceed to make your matching better?

Find an augmenting alternating path. That is, find a path starting at an unmatched node consisting of edges alternating between matched and unmatched edges to another unmatched node. Switching the two sets of edges increases the size of the matching. As with bipartite graphs.

Question: How do you find such a path?

In bipartite graphs the graph matched edges can be directed and from (say) left to right and the unmatched from right to left, and one can simply find a directed path. But with non-bipartite graph, how do we direct the edges?

Let's just search from an unmatched node for now. Trying to find an augmenting alternating path.

Question: What happens?

Well, you might find an alternating path. Cool!

Question: What else might happen?

Well, consider the example where you have an alternating path to a 5-cycle, with a matched edge (call it the stem) to the entry point and two matched edges on the cycle. Now, you go around the cycle and hop over the matched edge (stem) and whip out to another unmatched node. This is not an feasible alternating path. On the other hand, if an unmatched node is incident to any node on this cycle there is an alternating path. We sort of need to know which way to go around the cycle to find the alternating path.

Question: Can this get bad, in terms of searching?

We could build up an example, with a string of these where the direction around the cycles get to be a mess.

Question: How do we proceed?

Switch the matching on the stem. Collapse the cycle. Now, the internal matching can be changed so that any node is unmatched. Thus, the collapsed blossom is a free node.

Question: Does this help?

We state the following lemma regarding M which is the matching consisting of the original matching modified by switch the path to the blossom, and a blossom B that consists of an alternating cycle.

Lemma 18.1 *If M/B is a maximum matching of G/B if and only if M is a maximum matching of G .*

Question: Proof? One direction?

If there is a bigger matching N in G/B then the matching can be extended to a matching of G that has $(|B| - 1)/2$ more edges just as M has $(|B| - 1)/2$ more edges. Thus, M is not maximum if M/B is not maximum.

Question: Proof, the other way?

If M is not maximum. Then there is an augmenting path P between unmatched nodes u and v . If it does not involve B then M/B is not maximum. But, say u is not on B , and w is the first vertex on B that on the path, let Q be the portion of this path.

Question: Is Q an augmenting cycle for M/B in G/B ?

Yes. Thus, M/B is not maximum.

Question: Do we have an algorithm?

We start from an unmatched vertex. Do alternating search. First unmatched edges, then jump to mates. If one ever sees an edge at the same level, you found a blossom (it must be an odd cycle). So, we flip stem path, and collapse blossom. Number of nodes goes down by one. We restart search.

By induction with the lemma above. If the current matching is not optimal, we will eventually find an augmenting path.

Question: Running time?

At most n augmenting paths, at most n blossom collapses per augmenting path search, $O(m)$ time per blossom collapse. $O(n^2m)$ time algorithm.

18.2 Minimum cost perfect matching.

Question: What is a linear program?

$$\min \sum_e x_e c_e.$$

$$\sum_{e=(u,v)} x_e = 1.$$

$$\forall S \subseteq V, |S| \text{ odd and bigger than } 1 \quad \sum_{e \in E(S, \bar{S})} x_e \geq 1. \quad (18.1)$$

The last constraints (the odd sets constraints) certainly holds for perfect matchings. And we need it to prevent fractional solutions. On the other hand, we don't yet know if it is enough.

Question: What do we do now?

We take the dual.

$$\begin{aligned} \max \sum_S p_S + \sum_i p_i. \\ \sum_{e \in E(S, \bar{S})} p_S + p_u + p_v \leq c_e. \end{aligned} \tag{18.2}$$

Question: Ok? So?

Let's find a dual-primal feasible solutions that obey complementary slackness. A tight edge with respect to a dual solution is an edge where equation 18.2 holds with equality. In the primal x_e can only be positive on tight edges. In the dual p_S can only be positive on subsets S where the odd set equations 18.1 hold with equality.

Question: Initial solutions?

All x_e 's are 0, and all p_u and p_S are all 0. Infeasible primal (all nodes not matched), non-optimal dual.

Question: Adjust solutions maintaining complementary slackness?

Start at unmatched node. Raise p_u until there is one or more tight adjacent edges in this (developing) dual solution. If a tight edge leads to an unmatched node, match this edges (i.e., set x_e to 1.)

If all the tight edges correspond to matched nodes, they are matched along tight edges (complementary slackness.) Follow these nodes to get a set S .

To continue, for S , raise p_u on all even levels, and lower on all odd levels until new edge is tight, then add it and matched node, and continue. The lowering and raising keeps all edges in the current tree tight.

If one ever gets a tight unmatched edge or a matched edge between nodes on levels of same parity, we find a blossom. We collapse it and start all over.

Question: Why is it alright to raise p_u for unmatched nodes?

Well, it isn't unless in the end we get an augmentation? But we either do, or we get a blossom and we change the matched edges on the stem path accordingly and u gets matched.

Question: In the collapsed graph, we start raising $p_{v'}$ where v' is a collapsed blossom. what does this correspond to?

It is the p_B for the subset in the blossom which (is inductively) of odd size.

Question: Are we done?

Sure. We maintain complementary slackness after each search. After n collapses, we must find an augmenting path or no perfect matching exists.

Question: Running time?

The search needs to find the minimum δ to

Exercise 1: Sketch an implementation of this algorithm and bound

its running time.

Question: Structural result?

Sure, we showed there is an optimal integral solution the the linear programming formulation of perfect matching with odd set constraints.